REQUIREMENTS ANALYSIS AND SPECIFICATION

By

ANIL KUMAR DUDYALA

Assistant Professor, Dept of CSE

NIT Patna.

Source: Rajib Mall

Requirements Analysis and Specification

- Many projects fail:
  - Because they start implementing the system.
  - Without determining whether they are building what the customer really wants.

WHY BOTHER ABOUT REQUIREMENTS

– Improper requirements increase the no. of iterative changes required during life cycle phases.

– Incorrect requirements lead to huge costs.

– If not bothered leads to customer dissatisfaction and may also lead to legal battles.

GOAL OF REQUIREMENTS SPECIFICATION

– To clearly understand the customer requirements.

– Systematically organize the requirements into specification document.

– The output is called Software Requirements Specification (SRS).

WHO DOES THIS?

– System Analyst

– The person who gathers the information from the customer, analyzes and conceptualize it and projects it in an understandable way.

– During analysis the inconsistencies, anomalies and incompleteness are removed.

Qualities of System Analyst

– Some desirable attributes of a good system analyst:

– Good interaction skills,

– Imagination and creativity,

– Experience.

WHAT NEXT?

– Once the SRS is done it is given to the customer for review.

– When the customer agrees to it, it forms the basis for all future development activities and serves as a contract document between the customer and the development organization.

ACTIVITIES IN REQUIREMENTS

– There are mainly two activities involved in requirements, they are,

 – Requirements gathering and analysis

 – Requirements specification.

REQUIREMENTS GATHERING AND ANALYSIS

– This activity is further divided into two tasks

  – Requirements Gathering

  – Requirements Analysis

– Requirements Gathering is not a simple task as it involves interacting with different type of customers who have different level of understandability of the software and work.

REQUIREMENTS GATHERING

– Requirements gathering would be little easier if there exists a working model.

– It would be tedious if it has to be collected for a totally new project. Here, it would be a test for the analysts creativity and experience.

– Requirements collected generally would be in bits and pieces which has to be integrated and put it into proper meaningful format.

REQUIREMENTS GATHERING Contd..

– Requirement gathering can be done in the following way

  – Studying the existing documentation

  – Interview

  – Task analysis

  – Scenario analysis

  – Form analysis

– Read the case study in page 111 of Rajib Mall.

Analysis of the Gathered Requirements

– Main purpose of requirements analysis:

  – *Clearly understand the user requirements,*

  – *Detect inconsistencies, ambiguities, and incompleteness.*

– Incompleteness and inconsistencies:

  – Resolved through further discussions with the end-users and the customers.

Inconsistent Requirement

– Some part of the requirement:

  – contradicts with some other part.

– Example:

  – One customer says turn off heater and open water shower when temperature > 100 C

  – Another customer says turn off heater and turn ON cooler when temperature > 100 C

Incomplete Requirement

– Some requirements have been omitted:

– Possibly due to oversight.

– Example:

– The analyst has not recorded:
   when temperature falls below 90 C

– *heater should be turned ON*

– *water shower turned OFF.*

Analysis of the Gathered Requirements (CONT.)

– Requirements analysis involves:

- Obtaining a clear, in-depth understanding of the product to be developed,

- Remove all ambiguities and inconsistencies from the initial customer perception of the problem.

Analysis of the Gathered Requirements (CONT.)

– It is quite difficult to obtain:

– A clear, in-depth understanding of the problem:

– *Especially if there is no working model of the problem.*

Analysis of the Gathered Requirements (CONT.)

– Experienced analysts take considerable time:

  – To understand the exact requirements the customer has in his mind.
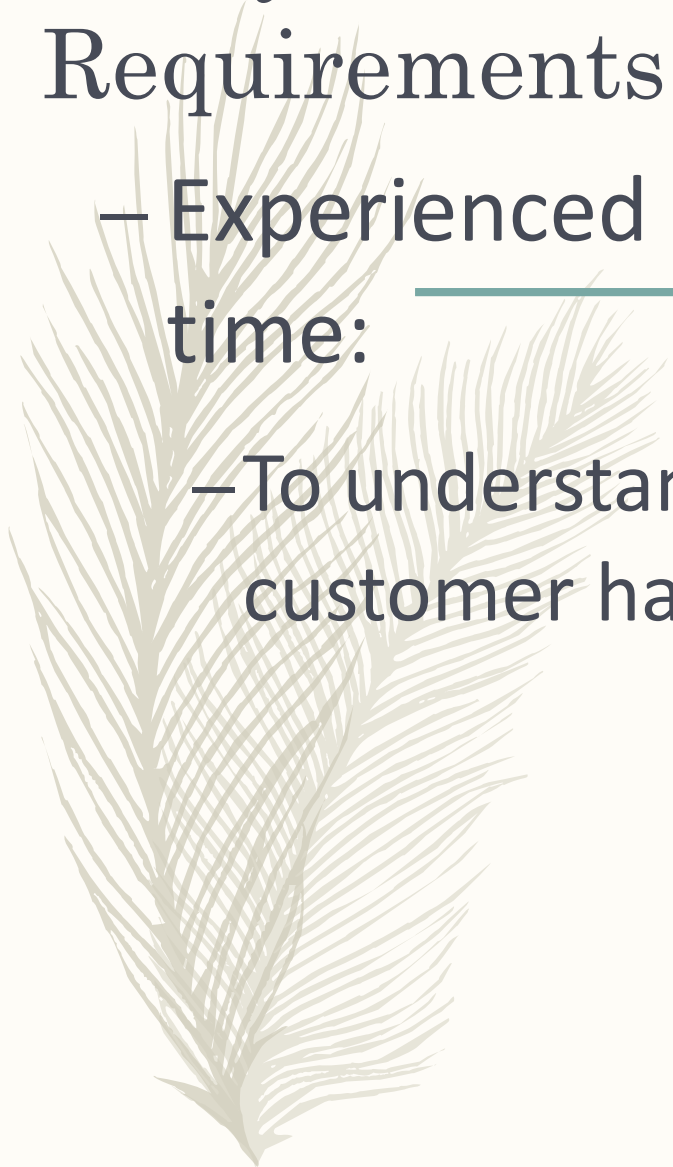
Analysis of the Gathered Requirements (CONT.)

– Experienced systems analysts know - often as a result of painful experiences ---

– Without a clear understanding of the problem, it is impossible to develop a satisfactory system.

Analysis of the Gathered Requirements(CONT.)

– Several things about the project should be clearly understood by the analyst:

  – What is the problem?

  – Why is it important to solve the problem?

  – What are the possible solutions to the problem?

  – What complexities might arise while solving the problem?

Analysis of the Gathered Requirements(CONT.)

- Some anomalies and inconsistencies can be very subtle:

  - Escape even most experienced eyes.

  - If a formal model of the system is constructed,
    - *Many of the subtle anomalies and inconsistencies get detected.*

Analysis of the Gathered Requirements(CONT.)

- After collecting all data regarding the system to be developed,
  - Remove all inconsistencies and anomalies from the requirements,
  - Systematically organize requirements into a Software Requirements Specification (SRS) document.

PROBLEMS TO BE SOLVED BY ANALYST

– Ambiguity

  – Using inappropriate words like upper grade, lower grade, etc

– Inconsistency

  – Contradictory values/data must not be present

– Incompleteness

  – Missing data or incomplete data.

SOFTWARE REQUIREMENT SPECIFICATION(SRS)

– Among all the documents produced during a software development, SRS is the toughest. Because, it cater to the needs of a wide variety of audience.

– Different people need SRS document for different purposes.

23

SRS Continued

– Different categories of users and their needs are as follows,

  – Users, customers and marketing people

  – Software developers

  – Test Engineers

  – User documentation writers

  – Project Managers

  – Maintenance Engineers

Software Requirements Specification

- Main aim of requirements specification:
  - Systematically organize the requirements arrived during requirements analysis.
  - Document requirements properly.

Software Requirements Specification

– The SRS document is useful in various contexts:

- Statement of user needs
- Contract document
- Reference document
- Definition for implementation

SRS Document (CONT.)

- The requirements at this stage:

  - Written using end-user terminology.

- If necessary:

  - Later a formal requirement specification may be developed from it.

Properties of a Good SRS Document

– It should be concise

  – and at the same time should not be ambiguous.

– It should specify what the system must do

  – and not say how to do it.

– Easy to change.,

  – i.e. it should be well-structured.

– It should be consistent.

– It should be complete.

Properties of a Good SRS Document (cont...)

- It should be traceable

  - You should be able to trace which part of the specification corresponds to which part of the design, code, etc and vice versa.

- It should be verifiable

  - e.g. "system should be user friendly" is not verifiable

Examples of Bad SRS Documents

– <u>Unstructured Specifications:</u>

  – Narrative essay --- one of the worst types of specification document:

    – *Difficult to change,*

    – *Difficult to be precise,*

    – *Difficult to be unambiguous,*

    – *Scope for contradictions, etc.*

Examples of Bad SRS Documents

- Noise:

  - Presence of text containing information irrelevant to the problem.

- Silence:

  - aspects important to proper solution of the problem are omitted.

Examples of Bad SRS Documents

- Overspecification:
  - Addressing "how to" aspects
  - For example, "Library member names should be stored in a sorted descending order"
  - Overspecification restricts the solution space for the designer.
- Contradictions:
  - Contradictions might arise
    - *if the same thing described at several places in different ways.*

Examples of Bad SRS Documents

- Ambiguity:
  - Literary expressions
  - Unquantifiable aspects, e.g. "good user interface"

- Forward References:
  - References to aspects of problem
    - *defined only later on in the text.*

- Wishful Thinking:
  - Descriptions of aspects
    - *for which realistic solutions will be hard to find.*

IMPORTANT ASPECTS OF SRS

– SRS normally contains three important parts

– Functional requirements

– Non-functional requirements

– Goals of implementation

Functional Requirements

– Functional requirements describe:

– A set of high-level requirements

– Each high-level requirement:

– *takes in some data from the user*

– *outputs some data to the user*

– Each high-level requirement:

– *might consist of a set of identifiable functions*

Functional requirements (CONT.)

– It is desirable to consider every system:

   – Performing a set of functions {fi}.

   – Each function fi considered as:

   – Transforming a set of input data to corresponding output data.

Input Data → **fi** → **Output Data**
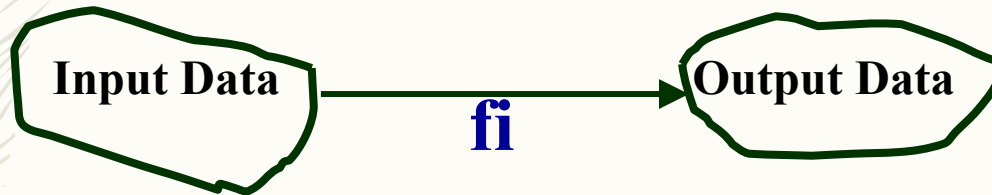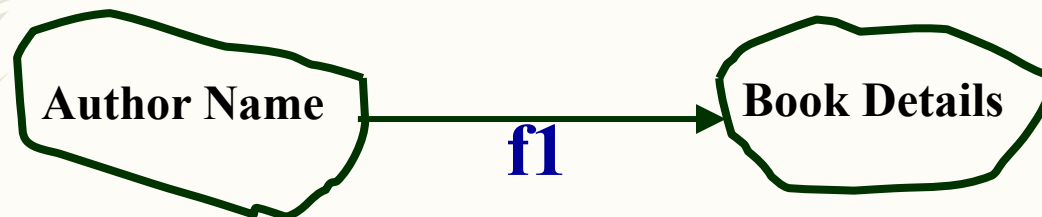
Example: Functional Requirement

– <u>F1:</u> Search Book

– Input:

– *an author's name:*

– Output:

– *details of the author's books and the locations of these books in the library.*

Functional Requirements

– For each high-level requirement:

–Every function is described in terms of:

–*Input data set*

–*Output data set*

–*Processing required to obtain the output data set from the input data set.*

Example Functional Requirements

- List all functional requirements
  - with proper numbering.
- Req. 1:
  - Once the user selects the "search" option,
    - *he is asked to enter the key words.*
  - The system should output details of all books
    - *whose title or author name matches any of the key words entered.*
    - *Details include: Title, Author Name, Publisher name, Year of Publication, ISBN Number, Catalog Number, Location in the Library.*

Req. 1:

– R.1.1:

  – Input: "search" option,

  – Output: user prompted to enter the key words.

– R1.2:

  – Input: key words

  – Output: Details of all books whose title or author name matches any of the key words/Error.

    – *Details include: Title, Author Name, Publisher name, Year of Publication, ISBN Number, Catalog Number, Location in the Library.*

  – Processing: Search the book list for the keywords

Nonfunctional Requirements

– Characteristics of the system which can not be expressed as functions:

- –Maintainability,
- –Portability,
- –Usability, etc.

Nonfunctional Requirements

– Nonfunctional requirements include:

  – Reliability issues,

  – Performance issues:

    – *Example: How fast the system can produce results*

      – so that it does not overload another system to which it supplies data, etc.

  – Human-computer interface issues,

  – Interface with other external systems,

  – Security, maintainability, etc.

Non-Functional Requirements

– Hardware to be used,

– Operating system

  – or DBMS to be used

– Capabilities of I/O devices

– Standards compliance

– Data representations

  – by the interfaced system

Goals of Implementation

– Goals describe things that are desirable of the system:

 – But, would not be checked for compliance.

 – For example,

  – *Reusability issues*

  – *Functionalities to be developed in future*

HOW TO IDENTIFY FR

– From an informal problem description document

– From a conceptual understanding of the problem

– The classification of high level requirement or other level is done with the help of domain expert.

HOW TO DOCUMENT FR

– Documentation is done by identifying the state at which the data is to be input to the system.

– Its input data domain

– The output data domain

– The type of processing to be done.

Example Functional Requirements

– List all functional requirements

   – with proper numbering.

– Req. 1:

   – Once the user selects the "search" option,

      – *he is asked to enter the key words.*

   – The system should output details of all books

      – *whose title or author name matches any of the key words entered.*

      – *Details include: Title, Author Name, Publisher name, Year of Publication, ISBN Number, Catalog Number, Location in the Library.*

Example Functional Requirements

- Req. 2:
  - When the "renew" option is selected,
    - *The user is asked to enter his membership number and password.*
  - After password validation,
    - *The list of the books borrowed by him are displayed.*
  - The user can renew any of the books:
    - *By clicking in the corresponding renew box.*

Req. 2:

- **R2.1:**
  - Input: "renew" option selected,
  - Output: user prompted to enter his membership number and password.
- **R2.2:**
  - Input: membership number and password
  - Output:
    - *list of the books borrowed by user are displayed. User prompted to enter books to be renewed or*
    - *user informed about bad password*
  - Processing: Password validation, search books issued to the user from borrower list and display.

Req. 2:

– <u>R2.3:</u>

- Input: user choice for renewal of the books issued to him through mouse clicks in the corresponding renew box.

- Output: Confirmation of the books renewed

- Processing: Renew the books selected by the in the borrower list.

ORGANIZATION OF SRS DOCUMENT

– Introduction.

– Functional Requirements

– Nonfunctional Requirements

  – External interface requirements

  – Performance requirements

– Goals of implementation

Handling complex logics

– The complex processing is analysed in two ways

  – Decision Trees

  – Decision tables

– A decision tree gives a graphic view of the processing logic involved in decision making.

– A decision table shows the decision-making logic and corresponding actions taken in a matrix form.

Decision Trees

– Decision trees:

   – Edges of a decision tree represent conditions

   – Leaf nodes represent actions to be performed.

– A decision tree gives a graphic view of:

   – Logic involved in decision making

   – Corresponding actions taken.

Example: LMS

– A Library Membership automation Software (LMS) should support the following three options:

  – New member,

  – Renewal,

  – Cancel membership.

Example: LMS

– When the <u>new member</u> option is selected,

  – The software asks details about the member:

    – *name,*

    – *address,*

    – *phone number, etc.*

Example(cont.)

- If proper information is entered,

  - A membership record for the member is created

  - A bill is printed for the annual membership charge plus the security deposit payable.
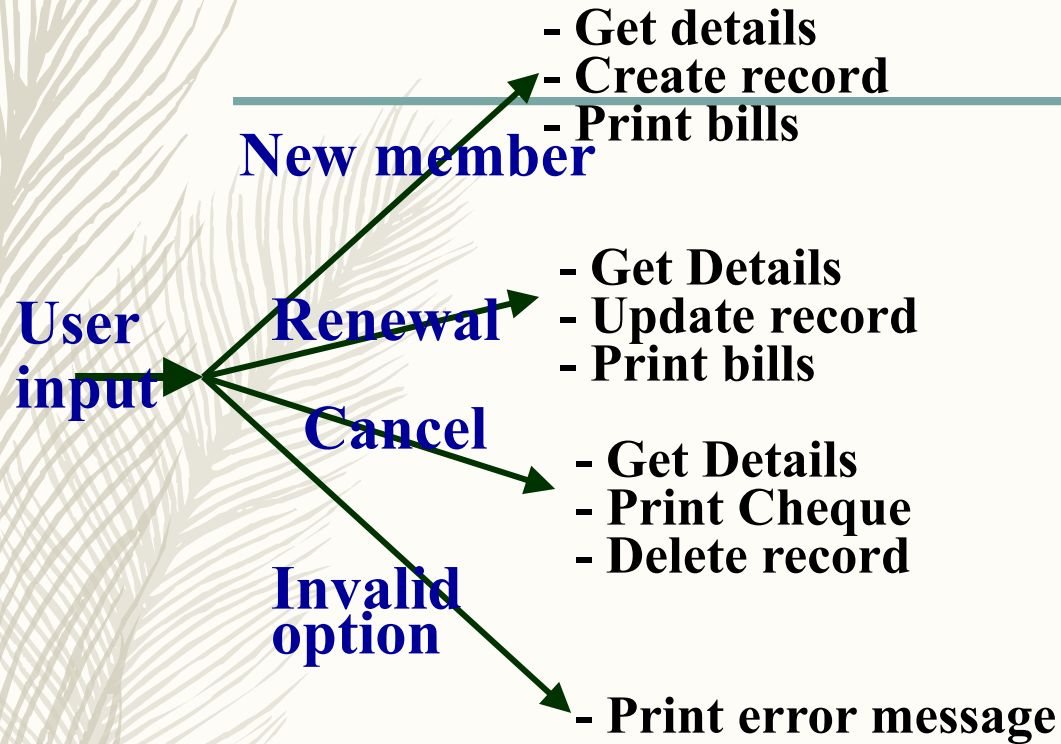
Example(cont.)

– If the [renewal](#) option is chosen,

    – LMS asks the member's name and his membership number

       – *checks whether he is a valid member.*

    – If the name represents a valid member,

       – *the membership expiry date is updated and the annual membership bill is printed,*

       – *otherwise an error message is displayed.*

Example(cont.)

- If the <u>cancel membership</u> option is selected and the name of a valid member is entered,

  - The membership is cancelled,

  - A cheque for the balance amount due to the member is printed

  - The membership record is deleted.

Decision Tree

User input

New member → - Get details
- Create record
- Print bills

Renewal → - Get Details
- Update record
- Print bills

Cancel → - Get Details
- Print Cheque
- Delete record

Invalid option → - Print error message

Decision Table

– Decision tables specify:

– Which variables are to be tested

– What actions are to be taken if the conditions are true,

– The order in which decision making is performed.

Decision Table

– A decision table shows in a tabular form:

  – Processing logic and corresponding actions

– Upper rows of the table specify:

  – The variables or conditions to be evaluated

– Lower rows specify:

  – The actions to be taken when the corresponding conditions are satisfied.

Decision Table

– In technical terminology,

– a column of the table is called a <u>rule</u>:

– A rule implies:

– *if a condition is true, then execute the corresponding action.*

Example:

- **Conditions**

Valid selection	NO	YES	YES	YES
New member	--	YES	NO	NO
Renewal	--	NO	YES	NO
Cancellation	--	NO	NO	YES

- **Actions**

Display error message	--			
Ask member's name etc.		--	--	
Build customer record		--		
Generate bill		--	--	--
Ask membership details			--	--
Update expiry date			--	--
Print cheque			--	--
Delete record				--

Comparison

– Both decision tables and decision trees

    – Can represent complex program logic.

– Decision trees are easier to read and understand

    – When the number of conditions are small.

– Decision tables help to look at every possible combination of conditions.

Formal Specification

- A formal specification technique is a <u>mathematical method</u> to:
  - Accurately specify a system.
  - Verify that implementation satisfies specification.
  - Prove properties of the specification.

Formal Specification

– Advantages:

– Well-defined semantics, no scope for ambiguity

– Automated tools can check properties of specifications

– Executable specification

Formal Specification

– Disadvantages of formal specification techniques:

- – Difficult to learn and use
- – Not able to handle complex systems
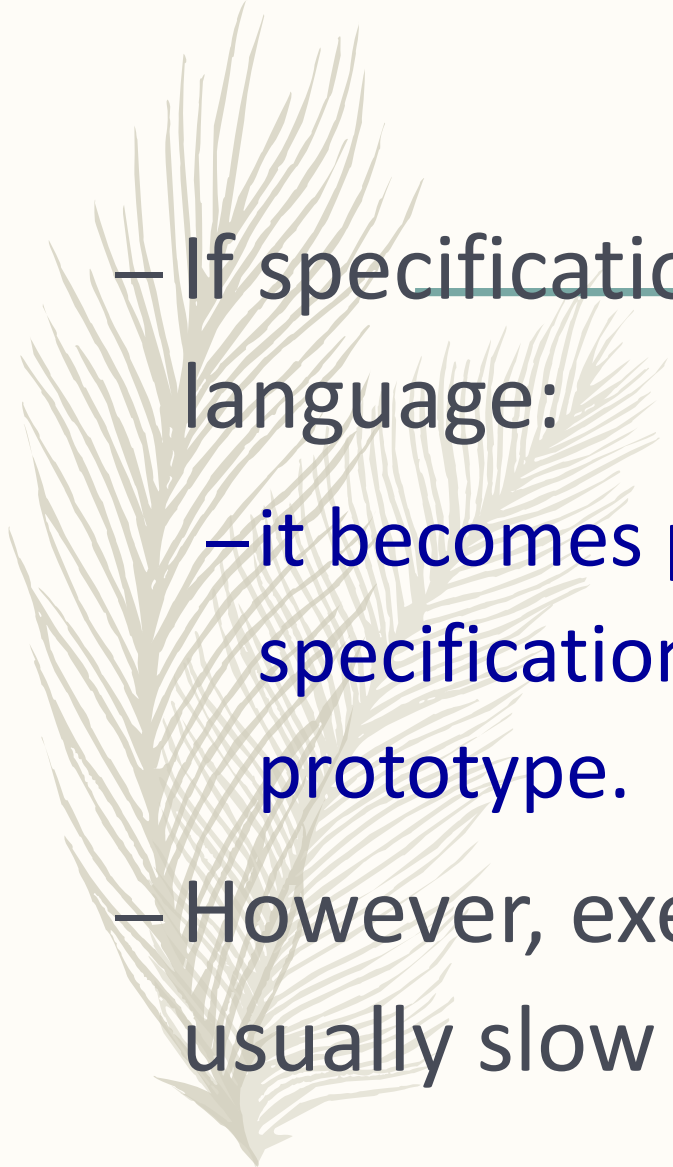
Formal Specification

– Mathematical techniques used include:

- Logic-based
- set theoretic
- algebraic specification
- finite state machines, etc.

Semiformal Specification

- Structured specification languages
  - SADT (Structured Analysis and Design Technique)
  - PSL/PSA (Problem Statement Language/Problem Statement Analyzer)
    - *PSL is a semi-formal specification language*
    - *PSA can analyze the specifications expressed in PSL*

Executable Specification Language

– If specification is expressed in formal language:

   – it becomes possible to execute the specification to provide a system prototype.

– However, executable specifications are usually slow and inefficient.

Executable Specification Language

– Executable specifications only test functional requirements:

– If non-functional requirements are important for some product,

– *The utility of an executable specification prototype is limited.*

4GLs

– 4GLs (Fourth Generation Languages) are examples of

  – executable specification languages.

– 4GLs are successful

  – because there is a lot of commonality across data processing applications.

4GLs

- 4GLs rely on software reuse
  - Where common abstractions have been identified and parameterized.
- Rewriting 4GL programs in higher level languages:
  - Result in upto 50% lower memory requirements
  - Also the programs run upto 10 times faster.

Summary

– Requirements analysis and specification

– An important phase of software development:

– Any error in this phase would affect all subsequent phases of development.

– Consists of two different activities:

– Requirements gathering and analysis

– Requirements specification

Summary

– The aims of requirements analysis:

  – Gather all user requirements

  – Clearly understand exact user requirements

  – Remove inconsistencies and incompleteness.

– The goal of specification:

  – Systematically organize requirements

  – Document the requirements in an SRS document.

Summary

– Main components of SRS document:

  – Functional requirements

  – Non-functional requirements

  – Constraints

– Techniques to express complex logic:

  – Decision tree

  – Decision table

Summary

–Formal requirements specifications have several advantages.

–But the major shortcoming is that these are hard to use.